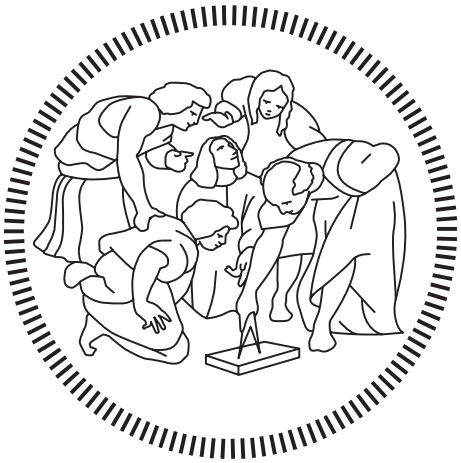
Politecnico di Milano

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Laurea Triennale – Ingegneria Informatica



Divisore Intero con Resto
Prova Finale Progetto di Reti Logiche

Tutor Accademico

Prof. Carlo Brandolese

Candidato

Nicola Cecere - 937506

Anno Accademico 2021 - 2022

Indice

Introduzione

La prova finale si concentra sulla progettazione di un divisore intero, avente come ingresso numeri binari naturali, rappresentati su 32 bit con intervallo di valori che varia da $[2^{32} - 1\ ;\ 0]$.

L'algoritmo utilizzato, si basa sul metodo della divisione lunga, secondo il quale:

per ogni coppia di polinomi $N(x)$ e $D(x)$, esistono unici altri due polinomi $Q(x)$ e $R(x)$ tali che $N(x) = D(x) \cdot Q(x) + R(x)$.

In particolare, siano N il dividendo, D il divisore, Q il quoziente ed R il resto, si può definire il metodo iterativo, per eseguire l'operazione di divisione proposto dal seguente pseudocodice:

```
1.  if(D == 0) {
2.    error();
3.  }
4.
5.  Q = 0
6.  R = 0
7.
8.  for(n = 31 ; n >= 0 ; n--) {
9.    R = R << 1
10.   R[0] = N[n]
11.   if(R >= D) {
12.     R = R – D
13.     Q[n] = 1
14.   }
15.   else {
16.     Q[n] = 0
17.   }
18. }
```

La rete effettua innanzitutto un controllo sul divisore in ingresso e si riporta in uno stato di errore nel caso quest'ultimo fosse nullo.

Successivamente, inizializza a zero il quoziente, il resto e inizia un ciclo con durata pari alla lunghezza della codifica binaria, nel nostro caso fissa a 32.

Nel ciclo, il resto della divisione viene shiftato di un bit, a partire dal meno significativo verso il più significativo e, viene inserito il bit ennesimo del dividendo.

In seguito, viene valutato, se il valore del resto è maggiore del divisore e, a seconda del risultato del confronto, si effettuano operazioni diverse.

Nel caso in cui il valore del resto sia maggiore, a questo viene sottratto il divisore e si inserisce un uno nel quoziente nella posizione ennesima; invece, se il valore del resto risulti minore del divisore, viene soltanto inserito uno zero nel quoziente nella posizione ennesima.

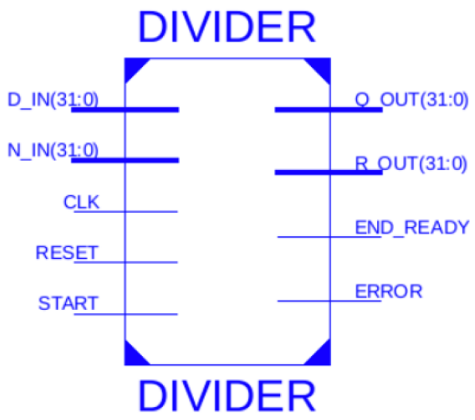
Infine, dopo aver ripetuto il ciclo per 32 iterazioni, se il divisore è diverso da zero, i registri del resto e del quoziente rappresenteranno su 32 bit il risultato corretto della divisione, al contrario presenteranno uno zero e lo stato di errore verrà indicato da un opportuno segnale.

Divisore Top Level

Interfaccia del sistema

Il divisore top level, ha in ingresso il dividendo N e il divisore D su cui effettuare il calcolo, il Clock per sincronizzare il circuito, il Reset per inizializzare il circuito al primo avvio e lo Start per indicare l'inizio di una nuova operazione.

I suoi segnali di uscita sono il quoziente Q e il resto R su cui è rappresentato il risultato del calcolo della divisione, il segnale di terminazione End_Ready, che indica la fine del processo di calcolo e la possibilità di iniziare una nuova operazione, l'indicatore di errore Error che segnala l'inserimento di un divisore nullo.



Input:

CLK:	in std_logic;
RESET:	in std_logic;
START:	in std_logic;
N_IN:	in std_logic_vector(31 downto 0);
D_IN:	in std_logic_vector(31 downto 0);

Output:

Q_OUT:	out std_logic_vector(31 downto 0);
R_OUT:	out std_logic_vector(31 downto 0);
ERROR:	out std_logic;
END_READY:	out std_logic;

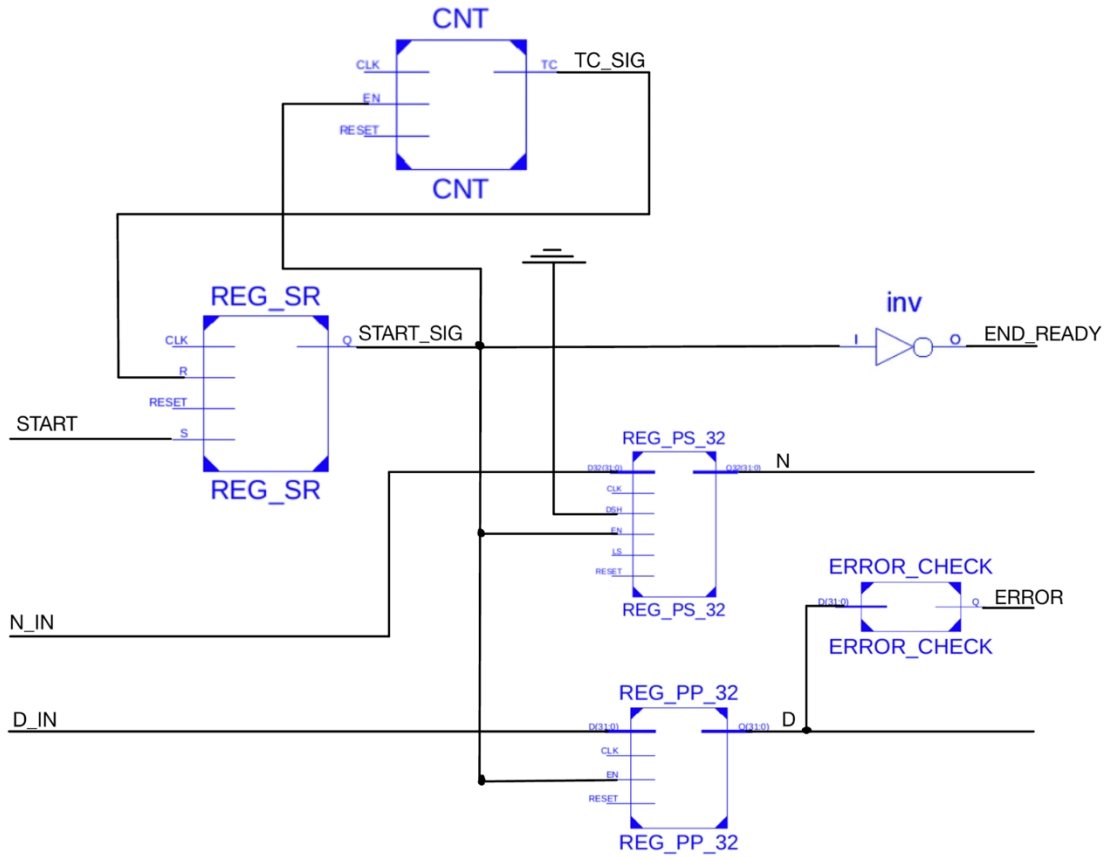
All'avvio del circuito è importante utilizzare il segnale di Reset con valore '1', affinché tutti i moduli all'interno del sistema possano essere inizializzati.

In seguito, per iniziare un'operazione di divisione, si pone il segnale di Start al valore '1' e si forniscono i valori del dividendo e del divisore, codificati su trentadue bit. I valori devono rimanere stabili per un ciclo di Clock, per essere campionati correttamente. Trascorsi trentatré cicli di Clock, dei quali trentadue per effettuare il calcolo e uno per campionare l'uscita dei risultati, si potranno visionare il resto e il quoziente della divisione codificati su trentadue bit.

Per effettuare una nuova operazione, è necessario che il segnale End_Ready mostri il valore '1'.

Architettura del sistema

Fase iniziale



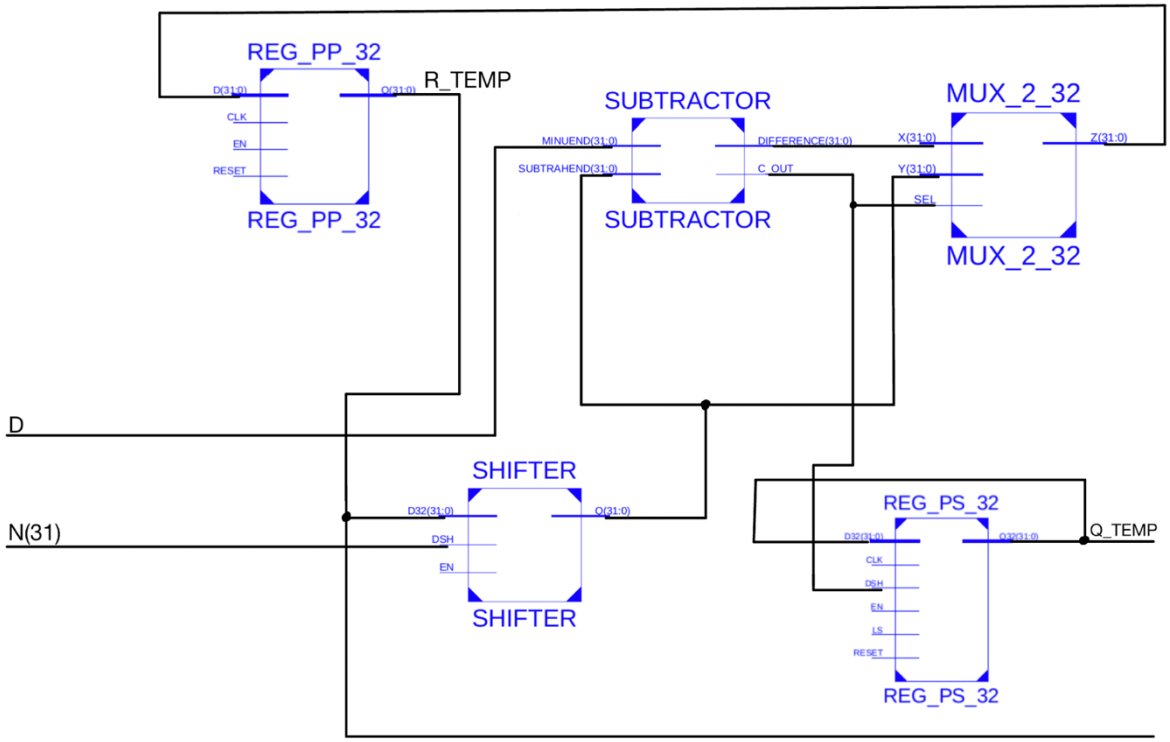
Il sistema nella fase iniziale di caricamento dati è composto da:

- un registro parallelo-parallelo che ha in ingresso il valore del divisore D;
- un registro parallelo-serie che ha in ingresso il valore del dividendo N;
- un contatore che genera il segnale di terminal count dopo 32 cicli di conteggio;
- un flip-flop SR che ha nell'ingresso della porta di Set S il segnale Start, preso dall'entrata del modulo di divisione, e nell'ingresso della porta di Reset R ha il segnale di terminal count generato dal contatore;
- un modulo asincrono collegato all'uscita del registro del divisore D che genera il segnale di errore in caso di divisore nullo.

L'uscita del flip-flop SR è utilizzata per alimentare i segnali di enable dei vari moduli interni e la sua negazione è collegata alla porta di End_Ready del sistema.

Inoltre, le combinazioni del segnale di Start e Tc sono utilizzate per alimentare opportunamente le porte di load-shift. Dunque, il contatore e il componente SR vengono utilizzati come logica di controllo del circuito, affinché i moduli siano abilitati al funzionamento nei momenti opportuni.

Fase di calcolo



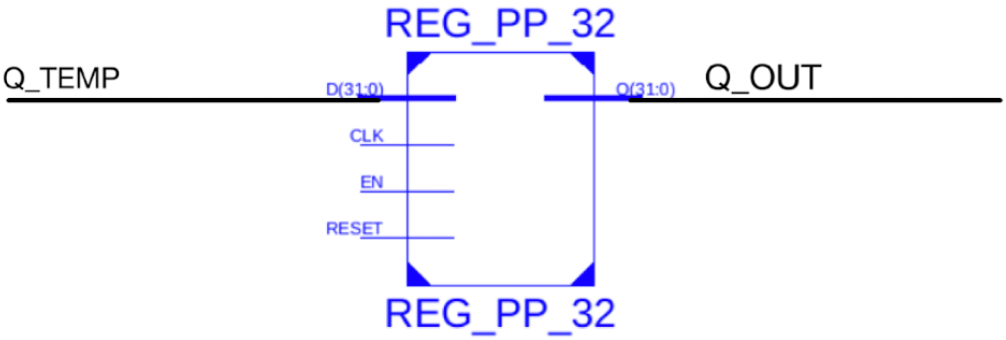
Il sistema nella fase di calcolo è composto da:

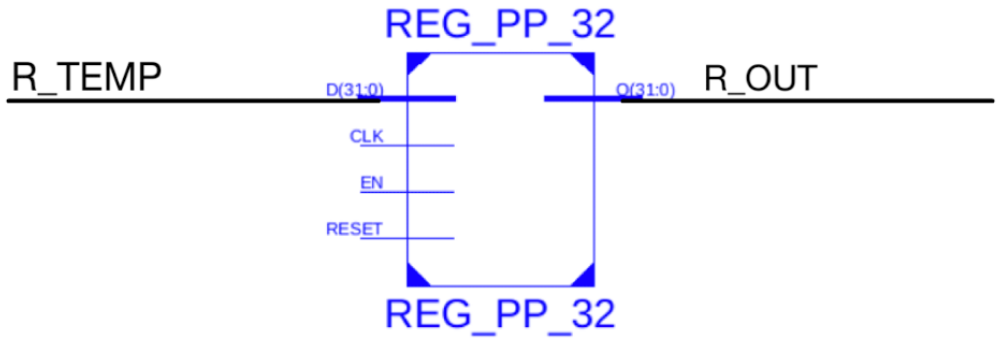
- un registro parallelo-parallelo che memorizza i valori temporanei del resto R;
- un registro parallelo-serie, che memorizza i valori temporanei del quoziente Q, in cui ad ogni ciclo di clock viene inserito un ‘1’o uno ‘0’ in base al risultato del confronto tra R e D;
- un sottrattore che effettua la sottrazione $R - D$ e ha in uscita il valore del riporto dell’operazione che permette di valutare se $R - D \geq 0$;
- uno shifter che, quando è abilitato, shifta il segnale proveniente dal registro contenente i valori temporanei di R, inserendo il bit più significativo dell’uscita del registro parallelo-serie N;
- un multiplexer a due ingressi da trentadue bit ciascuno, che riceve il segnale con il risultato di sottrazione del sottrattore, il segnale generato dallo shifter e che utilizza come operando di selezione il riporto d’uscita del sottrattore.

Il registro parallelo-parallelo, che memorizza il valore temporaneo del resto R, ha l’uscita collegata con l’ingresso dello shifter, in modo che possa essere svolta l’operazione $R = R \ll 1$, ed ha in ingresso il segnale selezionato dal multiplexer, per salvare il risultato della sottrazione se è maggiore di zero, oppure memorizzare il valore precedente shiftato di una posizione.

Il resto generato dal sottrattore è collegato anche all'ingresso del registro parallelo-serie, cosicché possa modificare il quoziente inserendo il valore opportuno.

Fase finale





Il sistema nella fase iniziale è composto da:

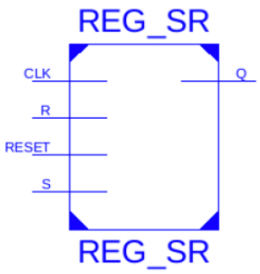
- un registro parallelo-parallelo che campiona il segnale di Q_TEMP solamente alla fine del calcolo della divisione se il segnale di errore ha valore pari a '0' ed è collegato alla medesima uscita del divisore per mostrare il risultato finale;
- un registro parallelo-parallelo che ha lo stesso funzionamento del primo registro ma con il segnale R_TEMP.

L'introduzione di questi due registri finali non è necessaria per il corretto funzionamento del divisore, ma è utile per mostrare soltanto il risultato finale dell'operazione e non l'intero processo di calcolo!

Infine, il Divisore Top Level contiene sei registri da trentadue bit, uno da otto bit e due da un bit, per un totale di duecento due flip-flop. Inoltre, il periodo minimo di Clock è di 58 ns, che rappresenta il ritardo massimo ed è causato dal sottrattore.

Moduli interni

Flip-Flop Set and Reset (SR)



Input:

CLK:	in std_logic;
RESET:	in std_logic;
S:	in std_logic;
R:	in std_logic;

Output:

Q:	out std_logic;

Il flip-flop SR riceve nell'ingresso della porta S il segnale di Start proveniente dall'esterno e su R il segnale interno di terminal count generato dal contatore.

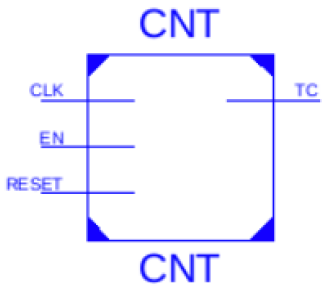
La sua uscita consente di abilitare i moduli che utilizzano un segnale di enable.

Inoltre, l'uscita del flip-flop attraversa un inverter e genera il segnale di End_Ready del divisore.

Il componente ha quindi una funzione di logica di controllo ed è utilizzato per definire lo stato del sistema.

Quando il valore memorizzato è pari a '0' il divisore è in uno stato di attesa, i suoi moduli interni sono disabilitati e la sua unica funzione è quella di mostrare i valori precedentemente calcolati, del resto R e quoziente Q. Invece quando il valore memorizzato è pari a '1' il sistema è in fase di elaborazione del risultato.

Contatore



Input:

CLK:	in std_logic;
RESET:	in std_logic;
EN:	in std_logic;

Output:

TC:	out std_logic;

Il contatore ha in ingresso un segnale di Enable ed è costruito con un registro parallelo-parallelo e un sommatore carry-look-ahead a otto bit.

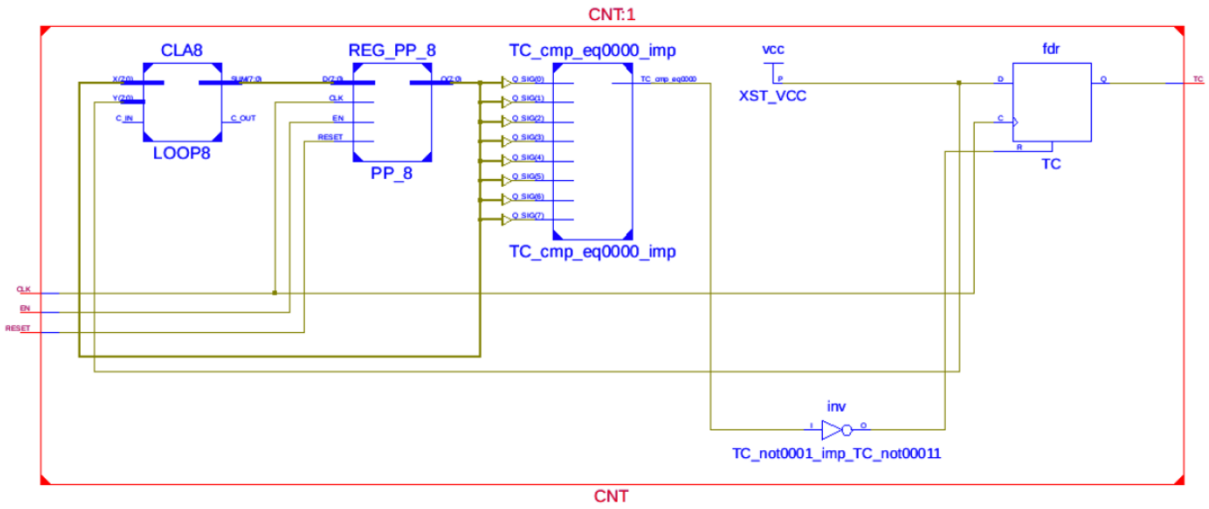
La sua funzione quando è abilitato, è quella di contare da ‘0’ al valore “31”, affinché si possa comprendere quando il ciclo di calcolo sia terminato e si siano svolti tutti i passaggi previsti dallo pseudo-codice.

Altrimenti se il contatore è disabilitato, questo rimane in attesa del segnale di Reset sul valore contato corrente.

In uscita è presente un segnale di terminal count, calcolato da una rete combinatoria collegata al registro parallelo-parallelo che confronta il valore memorizzato con “00011111” (31 in codifica binaria) e che in caso di esito positivo produce un ‘1’.

Inoltre, il segnale di terminal count generato in uscita è sincrono con il segnale di CLK grazie all’inserimento di un flip-flop che memorizza il valore prodotto dalla rete combinatoria.

Il contatore utilizza un registro e un sommatore a otto bit, nonostante il valore “31” sia codificabile con cinque bit, per creare un’istanza di un CLA a otto bit già creato per il modulo di sottrazione e confronto.



Registro PP a 8 bit

Il registro è composto da otto flip-flop, riceve in ingresso la somma generata dal contatore e la sua uscita è collegata alla rete combinatoria che genera il segnale di terminal count.

Il valore memorizzato viene modificato solamente se il segnale di Enable è alto.

Infine, la sua uscita è anche diretta verso l’entrata del sommatore CLA che aggiorna il valore contato.

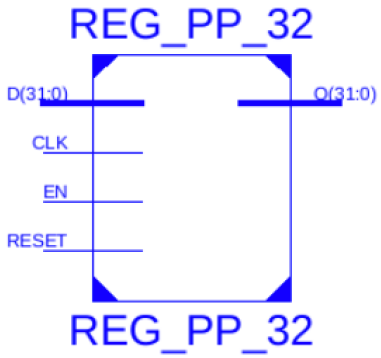
CLA a 8 bit

Il sommatore carry-look-ahead ha in ingresso tre segnali, ovvero, i due numeri da sommare e il resto iniziale.

In aggiunta presenta due uscite, delle quali, una rappresenta la somma e l'altra il resto finale. Il CLA è costruito secondo l'architettura ben nota formata da:

- un componente che produce i segnali di propagazione e generazione con $G(n) = X(n) \cdot Y(n)$ e $P(n) = X(n) + Y(n)$;
- un componente che combina i segnali di propagazione e generazione per anticipare il calcolo del riporto secondo la seguente logica: $C(i+1) = G(i) + P(i) \cdot C(i)$.
- n porte XOR a tre ingressi che ricevono $X(n), Y(n), C(n)$ e producono la somma $S(n)$.

Registro PP a 32 bit



Input:

CLK:	in std_logic;
RESET:	in std_logic;
EN:	in std_logic;
D:	in std_logic_vector(31 downto 0);

Output:

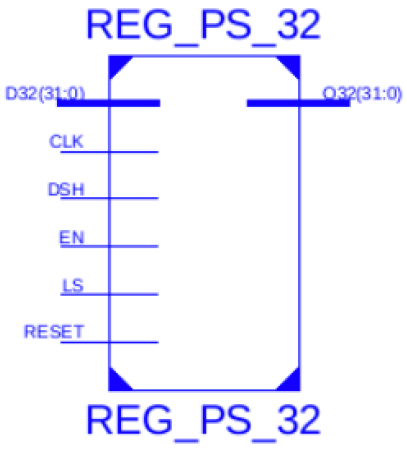
Q:	out std_logic_vector(31 downto 0);

Il registro parallelo-parallelo a trentadue bit è composto da trentadue flip-flop e presenta un ingresso dati, uno di abilitazione e l'uscita che mostra il valore memorizzato.

Le linee dati in ingresso sono connesse in parallelo agli ingressi dati dei flip-flop e le uscite sono le uscite degli stessi flip-flop: affinché la lettura dei dati avvenga in parallelo, tutti i flip-flop sono sincronizzati sullo stesso segnale di CLK.

Nel divisore ne sono istanziati quattro, di cui uno nella fase iniziale che campiona l'ingresso D, uno nella fase di calcolo che salva il valore temporaneo del resto R, due nella fase finale che memorizzano e mostrano i valori finali del quoziente Q e del resto R.

Registro PS a 32 bit



Input:

	CLK:	in std_logic;
	RESET:	in std_logic;
	D32:	in std_logic_vector(31 downto 0);
	DSH:	in std_logic;
	EN:	in std_logic;
	LS:	in std_logic;

Output:

	Q32:	out std_logic_vector(31 downto 0);

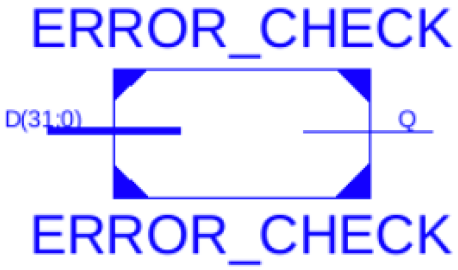
Il registro parallelo-serie ha due ingressi dati, di cui uno da trentadue bit e l'altro da un bit, un ingresso di abilitazione e riceve un segnale di controllo load-shift LS.

Quando il segnale LS vale '1', il registro memorizza in parallelo tutto il dato D32 in ingresso, come se si trattasse di un normale registro parallelo-parallelo. Quando LS vale '0', l'ingresso D32 viene sconnesso dai flip-flop ed il registro si comporta come uno shift register.

A differenza dei regolari registri parallelo-serie, invece di inserire un valore di default durante la fase di shift, questo modulo ne seleziona uno proveniente dall'ingresso. Inoltre l'uscita è parallela e non seriale, in modo tale da poter prevelare il dato memorizzato in uno solo fronte di CLK e non in 32.

Nel divisore ne sono istanziali due, uno nella fase iniziale che campiona l'ingresso N e uno nella fase di calcolo per salvare il valore temporaneo di Q.

Error Check



Input:

	D:	in std_logic_vector(31 downto 0);

Output:

	Q:	out std_logic;

L'error check ha in ingresso il valore di D, memorizzato nel registro parallelo-parallelo, e l'uscita collegata direttamente con la porta Error del divisore esterno.

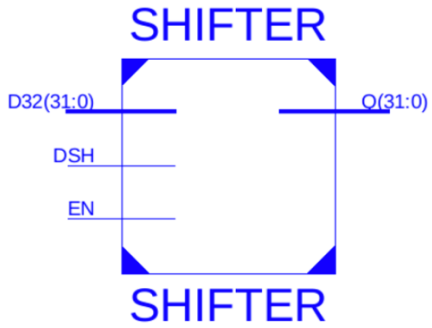
Quando l'ingresso è pari a '0' l'uscita del componente assume valore '1', invece quando l'ingresso ha un qualsiasi valore diverso da '0' l'uscita assume valore '0'.

Il modulo non ha in ingresso né un segnale di Reset , né di CLK in quanto è caratterizzato da un'architettura combinatoria e quindi asincrona.

Il divisore gestirà l'errore completando la fase di elaborazione e non aggiornando i registri della fase finale poiché i valori non avrebbero nessun significato.

L'utilizzatore sarà avvertito immediatamente dell'errore, tramite l'apposito segnale collegato all'uscita del divisore, e potrà ricominciare con una nuova operazione solo quando il segnale di End_Ready assumerà valore '1'.

Shifter



Input:

	D32:	in std_logic_vector(31 downto 0);
	DSH:	in std_logic;
	EN:	in std_logic;

Output:

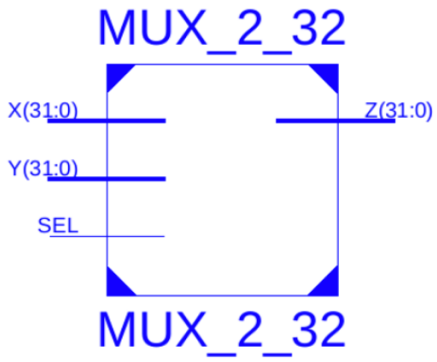
	Q:	out std_logic_vector(31 downto 0);

Lo shifter ha in ingresso il segnale da trentadue bit da shiftare, il valore da inserire nella posizione meno significativa del segnale shiftato e un segnale di abilitazione. Quando il valore del segnale di Enable è pari ad '1', lo shifter esegue l'operazione sopradescritta; invece quando assume valora pari a '0', l'uscita è collegata direttamente all'ingresso D32 senza nessuna modifica del dato.

Il modulo non ha in ingresso né un segnale di Reset , né di CLK in quanto è caratterizzato da un'architettura combinatoria e quindi asincrona.

Nell'architettura è istanziato solamente nella fase di calcolo, ha in ingresso della porta D32 il valore temporaneo del resto memorizzato nel registro parallelo-parallelo, mentre, ha in ingresso alla porta DSH il bit più significativo del segnale N memorizzato nel registro parallelo-serie presente nella fase iniziale.

Mux 2 ingressi da 32 bit



Input:

X:	in std_logic_vector(31 downto 0);
Y:	in std_logic_vector(31 downto 0);
SEL:	in std_logic;

Output:

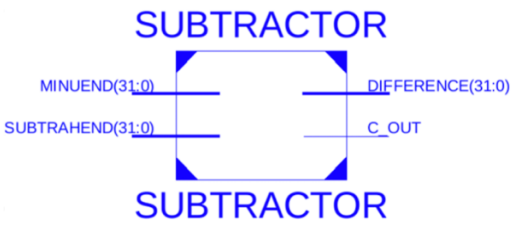
Z:	out std_logic_vector(31 downto 0);

Il multiplexer ha 2 ingressi dati da 32 bit e un segnale di selezione da un bit. Quando SEL assume valore '1' il multiplexer propagherà tramite la porta di uscita il segnale collegato all'ingresso X, altrimenti sarà propagato il segnale collegato all'ingresso Y.

Il modulo non ha in ingresso né un segnale di Reset , né di CLK in quanto è caratterizzato da un'architettura combinatoria e quindi asincrona.

Nell'architettura il multiplexer è istanziato solamente nella fase di calcolo, ha nell'ingresso della porta X il segnale di sottrazione prodotto dal sottrattore, nell'ingresso della porta Y il segnale del resto shiftato prodotto dallo shifter e come ingresso di SEL il resto della sottrazione. La sua uscita è riportata in ingresso al registro parallelo-parallelo che memorizza il valore temporaneo di R.

Sottrattore



Input:

MINUEND:	in std_logic_vector(31 downto 0);
SUBTRAHEND:	in std_logic_vector(31 downto 0);

Output:

DIFFERENCE:	out std_logic_vector(31 downto 0);
C_OUT:	out std_logic_vector(31 downto 0);

Il sottrattore ha nell'ingresso del minuendo un segnale da trentadue bit proveniente dall'uscita del registro parallelo-parallelo, presente nella fase iniziale e contente il valore del divisore D, nell'ingresso del sottraendo un segnale da 32 bit uscente dallo shifter che rappresenta il valore temporaneo di R.

Le sue uscite rappresentano la differenza ed il resto della sottrazione.

Il modulo utilizza un carry-look-ahead da trentadue bit composto da quattro CLA a 8 bit in serie, costruiti come mostrato nel paragrafo del contatore.

La scelta di dividere il CLA da trentadue bit in quattro parti è stata presa per avere un numero di porte contenuto a discapito di un leggero aumento del ritardo. Infatti, l'area del carry-look-ahead dipende in modo quadratico dal numero di bit dei segnali da sommare e ha un valore del ritardo costante.

In termini di porte logiche generiche abbiamo che la rete del CLA ha un'area $A = \frac{n^2+9n}{2}$, con n pari al numero di bit dei segnali, ed un ritardo $T = 4$ poiché costruita su quattro livelli di logica.

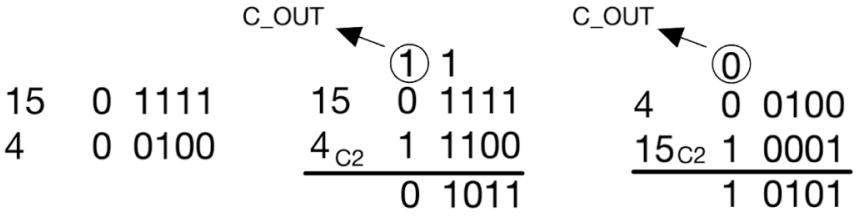
Il calcolo della sottrazione avviene secondo la seguente logica:

$$A - B = A + (-B) = A + B_{c2} = A + B_{c1} + 1$$

Quindi, al CLA da trentadue bit viene passato al primo ingresso il segnale del sottraendo, al secondo ingresso il segnale del minuendo negato tramite un inverter e al terzo ingresso il valore del resto pari a '1'.

Se il sottraendo è maggiore del minuendo il risultato di differenza sarà corretto e il resto in uscita avrà valore '1'. Invece se il sottraendo è minore del minuendo il risultato di differenza sarà codificato in complemento a 2 e il valore del resto in uscita sarà pari ad '0'.

Come esempio dimostrativo su 4 bit abbiamo che:



Questa regola, concernente il valore del resto, è sempre rispettata e può essere estesa ad un qualsiasi numero di bit. Infatti, nelle sottrazioni svolte ci sarà sempre un solo numero positivo ed un solo numero negativo, dunque, la somma del bit di segno è '1'.

Se il risultato della sottrazione è positivo, il bit di segno dovrà cambiare e necessiterà del resto di uscita pari a '1'. Invece se il risultato è negativo, il bit di segno non dovrà cambiare e il resto in uscita avrà valore '0'.

Di conseguenza, dato che il valore assunto dal resto della sottrazione è esplicativo, sul confronto $R \geq D$, questo segnale viene usato come scelta di selezione del multiplexer e come valore da inserire nel registro parallelo-serie che memorizza il risultato parziale di Q.

Test bench

In questo capitolo, viene mostrato quali test sono stati effettuati per verificare il buon funzionamento del Divisore Top Level.

I test sviluppati adottano un approccio “black-box” in cui i casi di test sono determinati dalla specifica e non dal codice scritto.

È stato fatto un partizionamento sistematico del dominio di input in modo tale da testare almeno un elemento rappresentativo di ogni partizione e i valori limiti.

N < D

1. START <= '1';
   N_IN <= (2 => '1', others => '0'); -- 4
   D_IN <= (2 => '1', 1 => '1', 0 => '1', others => '0'); -- 7
   Risultato → $R = 4,\ Q = 0$

2. START <= '1';
   N_IN <= (others => '0'); -- 0
   D_IN <= (3 => '1', others => '0'); -- 8
   Risultato → $R = 0,\ Q = 0$

3. START <= '1';
   N_IN <= (31 => '0', others => '1') ; -- 2.147.483.647
   D_IN <= (others => '1'); -- 4.294.967.295
   Risultato → $R = 2.147.483.647,\ Q = 0$

4. START <= '1';
   N_IN <= (0 => '0', others => '1') ; -- 4.294.967.294
   D_IN <= (others => '1'); -- 4.294.967.295
   Risultato → $R = 4.294.967.294,\ Q = 0$

N > D

1. START <= '1';
   N_IN <= (2 => '1', others => '0'); -- 4
   D_IN <= (1 => '1', 0 => '1', others => '0'); -- 3
   Risultato → $R = 1,\ Q = 1$

2. START <= '1';
   N_IN <= "00000000000000000000000000011100"; -- 28
   D_IN <= "00000000000000000000000000000101"; -- 5
   Risultato → $R = 3,\ Q = 5$

3. START <= '1';
   N_IN <= (others => '1'); -- 4.294.967.295
   D_IN <= (1 => '1', others => '0'); -- 2
   Risultato → $R = 1,\ Q = 2147483647$

4. START <= '1';
   N_IN <= (9 => '1' ,8 => '1, 6 => '1, 5 => '1' , 3 => '1', 2 => '1', others => '0'); --876
   D_IN <= (6 => '1' ,3 => '1', 2 => '1', 0 => '1', others => '0'); --71
   Risultato → $R = 24,\ Q = 12$

D = 0

1. START <= '1';
   N_IN <= (others => '1'); -- 4.294.967.295
   D_IN <= (others => '0'); -- 0
   Risultato → $R = 0,\ Q = 0$, ERROR = 1

2. START <= '1';
   N_IN <= (others => '0'); -- 0
   D_IN <= (others => '0'); -- 0
   Risultato → $R = 0,\ Q = 0$, ERROR = 1

N = D

1. START <= '1';
   N_IN <= (22 => '1', 21 => '1', 19 => '1',18 => '1', 15 => '1', 14 => '1', 13 => '1',
   10 => '1', 9 => '1', 7 => '1', 6 => '1', 4 => '1', 2 => '1', 1 => '1',
   others => '0'); --7136982
   D_IN <= (22 => '1', 21 => '1', 19 => '1',18 => '1', 15 => '1', 14 => '1', 13 => '1',
   10 => '1', 9 => '1', 7 => '1', 6 => '1', 4 => '1', 2 => '1', 1 => '1',
   others => '0'); --7136982
   Risultato → $R = 0,\ Q = 1$

2. START <= '1';
   D_IN <= (others => '1'); -- 4.294.967.295
   N_IN <= (others => '1'); -- 4.294.967.295
   Risultato → $R = 0,\ Q = 1$